Rekonfigurierbare Beschleuniger in der Welt von allgemeinen Rechenaufgaben[1]

Tobias Kenter[2]

Abstract: Rekonfigurierbare Beschleuniger, sogenannte FPGAs, sind schon lange als eine vielversprechende Architektur bekannt, die durch Anpassung an vielfältige Aufgaben hohe Rechenleistung und Effizienz erreichen kann. Dennoch werden sie bisher kaum für allgemeine Rechenaufgaben, sondern vorwiegend für Spezialaufgaben eingesetzt. In der vorgestellten Dissertation analysieren wir diese Diskrepanz und arbeiten Möglichkeiten und Herausforderungen, insbesondere bei der produktiven Anwendungsentwicklung für FPGAs, heraus. Die eigenen Beiträge dieser Arbeit konzentrieren sich vorwiegend auf den Bereich von FPGA-Overlay-Architekturen und kombinieren analytische Untersuchungen mit konkreten Lösungsansätzen und Implementierungen. So wird einerseits erstmalig der Performance-Unterschied zwischen Overlay-Lösungen und vollständig problemspezifischen FPGA-Schaltungen umfassend quantifiziert. Andererseits stellen wir automatisierte Übersetzungswerkzeuge für ein solches Overlay, verbesserte Laufzeitsysteme für Beschleuniger und eine höchst genaue Stereokorrespondenzberechnung mit FPGA-Beschleunigung vor. Vielbeachtete aktuelle Trends beim kommerziellen Einsatz rekonfigurierbarer Beschleuniger unterstreichen die hohe Relevanz unserer Arbeiten.

1 Einführung

Der Bedarf an immer höherer Rechenleistung für wachsende und neu aufkommende Rechenlasten und Märkte ist eine Herausforderung für die Rechnerarchitektur. Grenzen bei der Leistungsaufnahme und sinkende Geschwindigkeitszuwächse durch größere und zunehmend parallele Prozessoren machen Architekturinnovationen notwendig. Durch Effizienzsteigerungen und individuell angepasste Parallelität können spezialisierte Beschleuniger dazu beitragen die Rechenleistung für bestimmte Rechenlasten zu erhöhen. Ohne Programmierbarkeit fehlt ihnen jedoch die Flexibilität für allgemeine Rechenaufgaben (General-purpose computing) und sie können somit nicht davon profitieren, Kosten auf verschiedene Nutzungsszenarien und Märkte zu verteilen. Die Architektur von FPGAs (Field Programmable Gate Arrays), einer bestimmten Variante programmierbarer Logikbausteine, kombiniert Programmierbarkeit mit einem hohen Potenzial zur Spezialisierung für verschiedene Rechenlasten.

Das größte Hindernis zu breiterer Nutzung von FPGAs als Rechenbeschleuniger ist der Mangel an produktiven Formalismen und Werkzeugen für Anwendungsentwickler. Für spezielle Rechenlasten, deren Performance- oder Effizienzziele anders nicht erreicht werden können, wurden mit hohem Aufwand technisch ausgefeilte Lösungen mit FPGAs entwickelt [He07]. Für allgemeinere Rechenaufgaben kann ein solcher Aufwand häufig nicht

[1] Englischer Titel der Dissertation: „Reconfigurable Accelerators in the World of General-Purpose Computing“
[2] Universität Paderborn, kenter@uni-paderborn.de

geleistet werden. Dies gilt insbesondere, wenn die Zielarchitektur nicht, wie es beim Aufkommen von Grafikkarten (GPUs) als Beschleuniger ausnahmsweise der Fall war, bereits in sehr vielen Systemen vorhanden ist. Die FPGA-Hersteller und -Forschungsgemeinde haben einen hohen Aufwand betrieben, die Zugänglichkeit und Produktivität von Programmiermodellen, -sprachen und -werkzeugen zu verbessern. In dieser Dissertation analysieren wir, wie drei komplementäre Ansätze sich gegenseitig so ergänzen können, dass ein Durchbruch für die breite Nutzung von FPGAs als Rechenbeschleuniger möglich ist. Wir konzentrieren uns dabei auf übergelagerte Architekturen auf FPGAs (im Folgenden kurz Overlays). Solche Overlays sind ein vielversprechender Ansatz, die Produktivität bei der FPGA-Nutzung zu erhöhen, waren aber in verschiedenen Aspekten bisher unzureichend verstanden. So stellen wir erstmals eine umfassende Analyse dazu vor, mit welchen Kompromissen bei der erzielten Rechenleistung der Produktivitätsgewinn von instruktionsprogrammierten Overlays einher geht. Zudem leisten wir wesentliche Beiträge zu Fragen der konkreten Codegenerierung für solche Zielarchitekturen, der automatisierten Partitionierung zwischen Beschleunigercode und Programmteilen, die auf einem konventionellen Prozessor verbleiben, sowie der Laufzeitintegration solcher gemischter Ausführungsmodelle.

Das Themengebiet der gemischten Ausführung zwischen konventionellen Hochleistungsprozessoren und FPGA-Beschleunigern betrachten wir darüber hinaus auch aus der Perspektive der Rechnerarchitektur. Eine engere Integration zwischen FPGA und Prozessor hat nämlich einerseits einen großen Einfluss auf die produktive Anwendungsentwicklung für solche Systeme, andererseits aber auch direkte Auswirkungen auf die erzielbare Performance und Effizienz. In dieser Arbeit analysieren wir das Potenzial derartiger Systemintegration und diskutieren, auf welche Weise und in welchem Rahmen wir dieses Potenzial vorhersagen können. Die Herausforderung dabei ist, dass die mit einer Architektur erzielbare Rechenleistung und systemspezifische Anwendungsimplementierungen voneinander abhängen. Nachdem die engere Systemintegration von Grafikbeschleunigern schon seit einiger Zeit verfolgt wird, greift die Industrie diesen Trend inzwischen auch für FPGAs auf.

Im weiteren Verlauf dieses Überblicks werden die insgesamt sieben Kapitel der Dissertationsarbeit in drei Abschnitten und einem Fazit aufbereitet. Zunächst fassen wir in Abschnitt 2 unsere Betrachtungen und Ergebnisse im Bereich von Overlay-Architekturen zusammen. In Abschnitt 3 stellen wir Arbeiten zu Übersetzungs- und Laufzeittechniken für das zuvor genauer betrachtete Overlay und darüber hinaus vor. Abschnitt 4 behandelt die Untersuchungen zur Systemintegration von rekonfigurierbaren Beschleunigern mit konventionellen Prozessoren.

2 Vergleich zwischen Overlay-Lösungen und voller Spezialisierung

In den grundlegenden Betrachtungen dieser Dissertation wird herausgearbeitet, wie die Nutzung von Overlay-Architekturen einen wichtigen Beitrag zur verbreiteten Nutzung von FPGAs für allgemeine Rechenaufgaben leisten kann. Im Vergleich zu hochoptimierten anwendungsspezifischen Bibliotheken ermöglichen Overlays durch geringeren Entwicklungsaufwand, deutlich mehr Anwendungen zu erschließen. Sie liefern somit ein Kaufar-

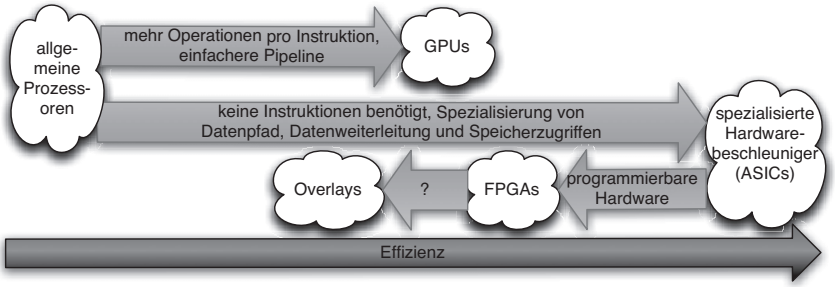


Abb. 1: Qualitative Illustration, wie sich architektonische Ansätze von Beschleunigern auf die Effizienz auswirken. Die Quantifizierung der Performanceauswirkungen von Overlays ist eine zentrale Forschungsfrage in dieser Dissertation.

gument für FPGAs in Systemen, die für ein breites und variables Anwendungsspektrum beschafft werden. Auch im Vergleich zur Synthese von FPGA-Designs aus Hochsprachencode, bei der insbesondere OpenCL als Beschreibungssprache vielversprechend ist, können Overlays den Entwicklungsaufwand deutlich reduzieren und vermeiden insbesondere auch die FPGA-typischen extrem langen Syntheselaufzeiten von einigen Stunden bis hin zu Tagen.

Wir unterscheiden bei den Overlay-Architekturen zwischen Prozessor-ähnlichen instruktionsprogrammierbaren Overlays und strukturell programmier- oder konfigurierbaren Overlays. Für beide Ansätze wurden diverse Architekturen vorgestellt, die ihre Effizienz aus verschiedenen Kombinationen von Parallelität, Pipelining und zielgerichteten Datenzugriffen und -wiederverwendung gewinnen. Diese werden im akademischen Umfeld bereits unter verschiedenen Gesichtspunkten wie Produktivität, Portierbarkeit und Skalierbarkeit untersucht. Insbesondere für instruktionsprogrammierbare Overlays war aber bis zu unseren Arbeiten weitgehend unklar, wie nah die Leistungsfähigkeit solcher Architekturen an die Leistung vollständig spezialisierter FPGA-Implementierungen herankommt. Abbildung 1 illustriert den Kontext dieser Forschungsfrage im Vergleich zu alternativen Beschleunigerarchitekturen.

Zur Beantwortung dieser Frage haben wir eine vielseitige Auswahl von Rechenaufgaben mit identischen Schnittstellen auf einem Overlay-basierten FPGA-System und mit weitreichend spezialisierten FPGA-Designs implementiert. Die Aufgaben sind dabei alle laufzeitintensiven Arbeitsschritte (häufig als Kernels bezeichnet) aus einer Anwendung zur Berechnung der Stereokorrespondenz, was wiederum der wichtigste und aufwändigste Zwischenschritt zur Tiefenberechnung aus einem Paar von Stereobildern ist. Indem wir dem bis vor kurzem qualitativ besten publizierten Algorithmus [Me11] in diesem Bereich folgen, erreichen wir einerseits einen von FPGA-spezifischen Optimierungen unbefangenen Vergleich zwischen Overlay und spezialisierten Kernels, andererseits aber auch die bis dato genaueste Stereoberechnung mit FPGA-Beschleunigung. Andere FPGA-Implementierungen in diesem Bereich (z.B. [Sh14, JM14, Ti14]) passen die Arbeitsschritte und ihre Abfolge unterschiedlich stark an die Zielarchitektur an und erreichen damit

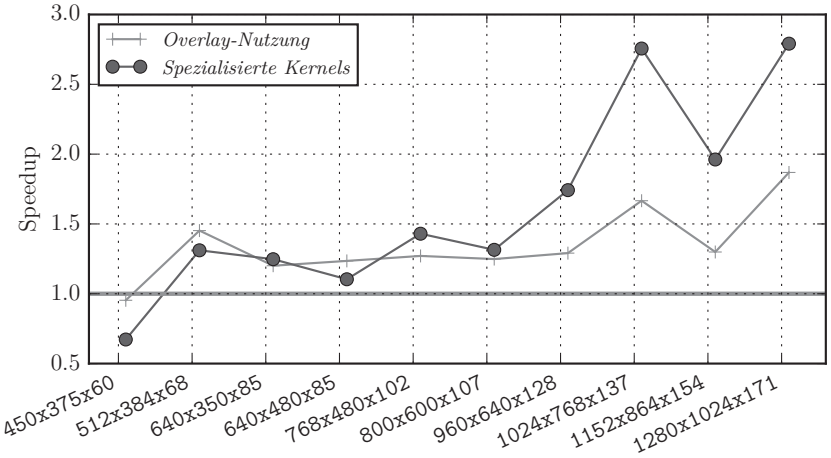


Abb. 2: Geschwindigkeitsvorteile der vollständig integrierten Berechnung der Stereokorrespondenz auf zwei Systemen mit FPGA Beschleunigern. In diesen Messwerten sind alle Zeiten zum Datentransfer und der Rekonfiguration enthalten, sodass die Lösung mit Overlays besonders bei kleinen Eingabegrößen gut abschneidet.

bei reduzierter Ergebnisqualität höhere Performanz oder geringeren Platzbedarf auf dem FPGA.

Bedingt durch die Verfügbarkeit und die Möglichkeiten die jeweiligen Ansätze mit Overlay und spezialisierten Kernels effektiv umzusetzen, wurden die beiden Ansätze auf zwei verschiedenen Zielplattformen realisiert. Die Convey HC-1 mit einem instruktionsprogrammierbaren FPGA-Overlay mit Vektorarchitektur und die Maxeler MPC-X Plattform mit einer eigenen Beschreibungssprache für hochgradig spezialisierte Datenflusskernels stellen zum jeweiligen Anschaffungszeitpunkt modernste Systeme mit performantem Serverprozessor und FPGA-Beschleuniger dar.

Auf beiden Zielplattformen wird die vollständig integrierte Berechnung der Stereokorrespondenz ausgeführt, indem die laufzeitintensiven Kernels auf den jeweiligen FPGA-Beschleuniger ausgelagert werden. Vorbereitende und verwaltende Arbeitsschritte verbleiben auf dem jeweiligen Hauptprozessor. Die in diesem Ausführungsmodell enthaltenen Laufzeiten für Datentransfers, Synchronisation und im Falle der spezialisierten Kerneldesigns auch Rekonfiguration begrenzen die erzielbare Performance. Dennoch überzeugen, in Abbildung 2 illustriert, beide Beschleunigerplattformen für die meisten Eingabegrößen der Anwendung durch Geschwindigkeitsvorteile gegenüber dem leistungsfähigen Hauptprozessor der Maxeler MPC-X Plattform.

Um die konzeptionell bedingten Performance-Unterschiede zwischen der Verwendung eines instruktionsprogrammierbaren Overlays und vollständig spezialisierten FPGA-Designs zu quantifizieren, mussten wir Einflüsse der konkreten Plattformen und ihrer Laufzeitumgebungen herausrechnen. In der Dissertation und in [KSP15] werden diese Schritte ausführlich erläutert. Anschließend kann gezeigt werden, dass die Verwendung des Overlays

im Schnitt etwa einen Faktor 3 weniger isolierte Kernelperformance liefert als vollständig an die Aufgabe angepasste FPGA-Implementierungen. Im Gegenzug sind für das Overlay die Laufzeiten der verwendeten Werkzeuge zur Übersetzung beziehungsweise Synthese um mehrere Größenordnungen kürzer und betragen nur Sekunden anstelle von Stunden oder gar Tagen. Die für einen Mehrwert notwendige hohe Produktivität wird aber nicht allein durch die Werkzeuglaufzeiten erreicht, sondern vor allem durch automatisierte Werkzeuge zur Code- oder Konfigurationsgenerierung für und Auslagerung auf Beschleuniger. Unsere Beiträge dazu werden im folgenden Abschnitt 3 diskutiert.

3 Übersetzungs- und Laufzeittechniken für FPGA-Beschleuniger

In verwandten Arbeiten wurden für verschiedene strukturell programmierbare oder konfigurierbare Overlays schnelle Codegenerierung und hoher Automatisierungsgrad der Werkzeuge bereits demonstriert [CS10, Fe11, Bi13]. Die Auswahl und Auslagerung geeigneter Kernels steht dabei entweder nicht im Vordergrund oder macht sich die enge Kopplung des rekonfigurierbaren Beschleunigers im Datenpfad eines eingebetteten Prozessors zunutze. Für die in Abschnitt 2 untersuchte Overlay-Architektur mit Vektorinstruktionen und die weniger eng gekoppelte Convey HC-1 Plattform war eine hohe Produktivität noch nicht demonstriert. Die verfügbaren Werkzeuge erforderten noch signifikanten Entwicklungsaufwand und schöpften weder die Leistungsfähigkeit noch die Vielseitigkeit des Overlays voll aus [KSP12].

Um die postulierten Produktivitätsvorteile kohärent zu demonstrieren, wollten wir daher für diese Plattform mit dem analysierten Vektor-Overlay Methoden und Werkzeuge präsentieren, die eine vollständig automatisierte Auslagerung auf den FPGA-Beschleunger ermöglichen und bessere Ergebnisse bei der Codegenerierung erzielen. Mit dem in [KVP14] vorgestellten Übersetzungssystem ist uns dies gelungen. Es baut auf der LLVM Compiler Infrastruktur[3] auf und kombiniert existierende und selbstentwickelte Analyse- und Transformationsphasen um zur Auslagerung geeignete Programmteile zu identifizieren und extrahieren. Anschließend generiert das System Beschleunigercode und orchestriert die notwendigen Datentransfers.

Zwischen den technischen Möglichkeiten der Overlay-Architektur und den existierenden Übersetzungswerkzeugen haben wir als wesentliche Lücke für die Produktivität die fehlende Codegenerierung für indirekt über Zeiger angesprochene Datenstrukturen ausgemacht. Im Sinne der Ergebnisqualität haben wir auch die Vektorisierung von äußeren Schleifen untersucht, die nach Vorläufern in der Zeit der Vektorgroßrechner auch auf aktuellen Prozessoren mit sogenannten SIMD-Instruktionen (Single Instruction Multiple Data) wieder aufgegriffen wurde [NZ08].

Neben der korrekten Funktionalität haben wir auch die Auswirkungen verschiedener Programmeigenschaften und Vektorisierungsmethoden anhand von systematisch variierten Schleifen untersucht. Wir unterscheiden zwischen verschachtelten Schleifen, die Vektorisierung der inneren oder äußereren Schleifen erfordern, oder unabhängig sind, zwischen

[3] http://llvm.org/

Tab. 1: Erreichte Performancegewinne als geometrisches Mittel für verschiedene Gruppen von Schleifen. Tiefgestellte Indizes zeigen die Größe der jeweiligen Gruppen. Identische Schleifen sind sowohl im Innen/Außen als auch im Passend/Transponiert Gruppenpaar enthalten.

	Innen	*Außen*	*Passend*	*Transponiert*	*Unabhängig*	Mittel
Array	6.51_{08}	6.67_{08}	13.10_{08}	3.31_{08}	2.05_{08}	4.46_{24}
Zeiger	2.00_{08}	2.01_{08}	2.62_{08}	1.52_{08}	0.88_{06}	1.52_{22}
Mittel	3.60_{16}	3.66_{16}	5.86_{16}	2.25_{16}	1.35_{14}	2.61_{46}

für die Zugriffsfolgen passenden oder transponierten Datenanordnungen und zwischen direkter Adressierung in Arrays und indirekter mit Zeigern. Wie in Tabelle 1 zusammengefasst, kann unser Ansatz für das Overlay für fast alle Gruppen von Schleifen Performancegewinne gegenüber der Ausführung auf einem konventionellen Serverprozessor erzielen. Die Adressierung und Datenanordnung, die wir hier als vom Programm vorgegeben betrachten, haben dabei deutlich größeren Einfluss auf die Performance als die Vektorisierungsmethode.

Eine weitere, hier nicht gezeigte Einflussgröße für das Auslagerungspotenzial ist die Schleifengröße selbst. In diesem Kontext haben wir gezeigt, dass es häufig vorteilhaft ist, konkrete Auslagerungsentscheidungen erst zur Programmlaufzeit zu treffen. Damit können, basierend auf Programmtransformationen zur Übersetzungszeit, wohlinformierte Entscheidungen über die Auslagerung getroffen werden, ohne dass das Programm vorher mit charakteristischen Eingabedaten vermessen werden muss. Dieser Ansatz stellte sich auch für das weitere Umfeld von Laufzeitsystemen [Lö16] für Beschleuniger als sehr vielversprechend heraus [Va14, Va16].

4 Systemintegration von Prozessoren und Beschleunigern

Neben dem Schwerpunkt auf Produktivität zur weiteren Verbreitung von FPGAs, müssen diese auch in etablierte Rechnersysteme und Märkte integriert werden und dazu architektonisch mit allgemeinen Prozessoren zusammenpassen und -arbeiten. Die engere Integration der Komponenten, beispielsweise in eine gemeinsame Speicherhierarchie, wie sie seit einigen Jahren im Bereich der Grafikbeschleuniger verfolgt wird, verspricht dabei nicht nur allgemein höhere Performance und einfachere Nutzung des Beschleunigers, weil zum Beispiel explizite Datentransfers wegfallen. Sie kann auch neue, viel engere Formen der Zusammenarbeit zwischen Prozessor und Beschleuniger ermöglichen. Eine weitblickende Entwurfsplanung für solche Architekturen ist dabei herausfordernd, weil das vollständige Performance-Potenzial einer Architektur nur in Kenntnis des am besten passenden Modells zur Programmpartitionierung und -ausführung bestimmt werden kann.

Zur Untersuchung dieser Fragestellung stellen wir daher ein abstraktes Modell zur Abschätzung von Rechenleistung von gekoppelten Prozessor-Beschleuniger Architekturen vor, das die gegenseitige Abhängigkeit zwischen Architekturen und Programmierentscheidungen berücksichtigt, indem für jede Architekturkonfiguration ein passendes Ausführungsmuster für eine Reihe von Benchmarkprogrammen gefunden wird [Ke11]. Die Ab-

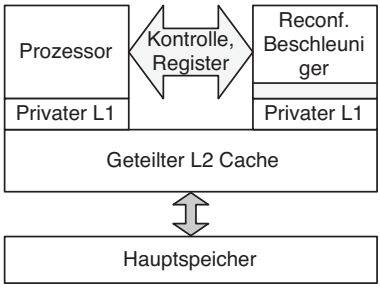
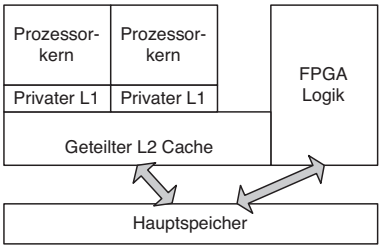


Abb. 3: Beispielarchitektur mit geteiltem L2 Cache und privaten L1 Caches. Die zweiteilige Schnittstelle des Beschleunigers in das System ist gelb hervorgehoben.

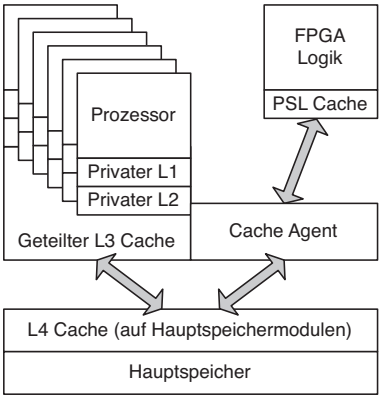
schätzung wird dabei automatisch anhand von Instruktionen und Datenzugriffen vorgenommen. Unser Ansatz ergänzt vorangehende Arbeiten [Sp09] um die Modellierung einer Cachehierarchie und eine bessere Berücksichtigung von Latenzen. Im Gegensatz zu Abschnitt 3 wird hier keine konkrete Codegenerierung vorgenommen. Damit werden Festlegungen auf bestimmte Eigenschaften eines rekonfigurierbaren Beschleunigers vermieden.

Wir erkunden mit dieser Methode den Entwurfsraum für Systeme, in die der Beschleuniger durch eine zweiteilige Schnittstelle integriert ist. Eine direkte Verbindung mit dem Prozessor ermöglicht eine feingranulare Synchronisation und den Austausch von Registerwerten, während weitere Daten über die Einbindung in eine gemeinsame Speicherhierarchie selbständig gelesen und geschrieben werden können. Abbildung 3 stellt eine mögliche Inkarnation dieses Architekturmusters dar. Zur Entwurfsraumexploration variieren wir die Aufteilung der Speicherhierarchie, Latenzen der Kontroll- und Registerschnittstelle, sowie das Potenzial des Beschleunigers selbst. Unsere Ergebnisse zeigen, dass die Speicherintegration mit möglichst enger Kopplung über einen geteilten Cache, sowie zusätzlichem privaten Speicher auf Seiten des Beschleunigers zentral für die Leistungsfähigkeit des Gesamtsystems ist. Dagegen stellte sich die Latenz der direkten Schnittstelle zum Prozessor als weniger wichtig heraus.

Aktuell scheinen kommerzielle Architekturen diesen Trend zu bestätigen. Beispielhaft illustrieren wir hier in Abbildung 4a die System-on-Chip (SoC) Xilinx Zynq Architektur für eingebettete Systeme, in der eine FPGA-Struktur als Beschleuniger mit einem ARM Cortex-A9 Prozessor gekoppelt wird. Der FPGA hat dabei Zugriff auf den gemeinsamen externen Hauptspeicher durch sogenannte hochperformante Schnittstellen und auf den geteilten L2 Cache durch die Beschleuniger-Kohärenz-Schnittstelle ACP (Accelerator Coherency Port). Ein Zugriff auf Prozessorregister ist dagegen nicht möglich, wohl aber können Interrupts auf dem Prozessor ausgelöst werden. Mit der CAPI (Coherent Accelerator Processor Interface) Schnittstelle bietet nun auch die IBM POWER8 Architektur für hochperformante Server-Prozessoren Unterstützung für Beschleuniger als Teil einer gemeinsamen Speicherhierarchie (Abbildung 4b). Ein Cache-Agent ist dafür im Auftrag des extern durch die PCIe Schnittstelle angeschlossenen Beschleunigers in die Daten- und Koheränzschnittstellen auf dem Prozessorchip integriert. Zum Konzept der Nutzung von FPGA-Beschleunigern mittels CAPI gehört darüber hinaus auch ein privater Cache auf dem FPGA selbst.



(a) Speicherhierarchie der Xilinx Zynq Architektur.



(b) Integration eines Beschleunigers mittels CAPI in die POWER8 Speicherhierarchie.

Abb. 4: Aktuelle Architekturen, die FPGA-Beschleuniger mit Prozessoren koppeln.

5 Fazit

Vielbeachtete aktuelle Entwicklungen und Ankündigungen, wie erstens die milliardenschwere Übernahme[4] von Altera, einem der beiden größten FPGA-Hersteller durch Intel, begleitet von der Ankündigung kombinierter Prozessor-FPGA Produkte, zweitens der großflächige Einbau von FPGAs in Microsofts Rechenzentren[5], und drittens, nach Veröffentlichung dieser Dissertation, die Ankündigung von FPGAs in der Amazon Cloud[6], zeigen mit wie viel Unterstützung rekonfigurierbare Beschleuniger gegenwärtig tatsächlich in Rechenzentren ankommen.

Diese Entwicklung wurde in dieser Dissertation aus architektonischer Sicht, insbesondere aber aus einer Produktivitätsperspektive, eingeordnet und weitere Chancen und Herausforderungen, auch für andere Marktsegmente, dargestellt. Die vorgestellten Ergebnisse zum besseren Verständnis von FPGA-Overlay Architekturen sowie die Werkzeuge und Laufzeitsysteme zur einfacheren Nutzung solcher Overlays und von FPGA-Beschleunigern im allgemeinen können einen entscheidenden Beitrag dazu leisten, dass der Einsatz von FPGAs nicht auf wenige publikumswirksame Erfolgsbeispiele beschränkt bleibt, sondern tatsächlich eine breite Nutzung für allgemeine Rechenaufgaben erfährt.

Eine besondere Qualität dieser Arbeit liegt dabei in der Kombination, auf der einen Seite von Konzepten und Analysen, die den Bereich des Rechnens mit rekonfigurierbaren Beschleunigern insgesamt voran bringen, auf der anderen Seite von konkreten technischen Lösungen, die nicht nur die konzeptuellen Arbeiten validieren, sondern auch für sich genommen im FPGA-Umfeld Anerkennung finden. So wird die Analyse zur Effizienz eines Overlay-Ansatzes begleitet von der bisher genauesten FPGA-gestützten Implementierung

[4] https://newsroom.intel.com/news-releases/intel-completes-acquisition-of-altera/
[5] https://www.wired.com/2016/09/microsoft-bets-future-chip-reprogram-fly/
[6] https://aws.amazon.com/de/ec2/instance-types/f1/

zur Berechnung von Stereokorrespondenz. Eine verbindende Bibliothek zur Speicherverwaltung und austauschbare Implementierungen der laufzeitintensiven Schleifen erlauben es darüber hinaus, die Anwendung auf zwei verschiedenen Beschleunigerplattformen mit unterschiedlichen Laufzeitumgebungen auszuführen. Im Bereich der Übersetzungstechniken für rekonfigurierbare Beschleuniger wird ebenfalls ein konkreter Werkzeugsatz zur automatisierten Auslagerung auf einen FPGA vorgestellt. Gleichzeitig werden aber auch Einflüsse der Daten- und Schleifenanordnung systematisch analysiert. Die Arbeiten zur Systemintegration von FPGA-Beschleunigern schließlich fokussieren sich auf analytische Beiträge. Die Übereinstimmungen mit den aktuellen Entwicklungen bei kommerziellen Produkten unterstreichen aber auch hier die praktische Relevanz.

Wesentliche Ergebnisse, die zu dieser Dissertation geführt haben, wurden in neun Konferenzbeiträgen, einem Workshopbeitrag und zwei Fachzeitschriftenartikeln präsentiert und veröffentlicht. Tobias Kenter ist Hauptbeitragender und erster Autor von sechs dieser zwölf Veröffentlichungen, darunter dem Artikel in einer der beiden wichtigsten Fachzeitschriften im Bereich des rekonfigurierbaren Rechnens [KSP15]. Die Bibliographie dieser Zusammenfassung kombiniert einen Teil dieser Veröffentlichungen mit einigen verwandten Arbeiten in den angesprochenen Teilgebieten.

Literaturverzeichnis

[Bi13] Bispo, João; Paulino, Nuno; Cardoso, João M. P.; Ferreira, João Canas: Transparent Runtime Migration of Loop-Based Traces of Processor Instructions to Reconfigurable Processing Units. Int. Journal of Reconfigurable Computing (IJRC), 2013, 2013.

[CS10] Coole, James; Stitt, Greg: Intermediate fabrics: virtual architectures for circuit portability and fast placement and routing. In: Proc. Int. Conf. on Hardware/Software Codesign and System Synthesis (CODES/ISSS). ACM, New York, NY, USA, S. 13–22, 2010.

[Fe11] Ferreira, R.; Vendramini, J.G.; Mucida, L.; Pereira, M.M.; Carro, L.: An FPGA-based heterogeneous coarse-grained dynamically reconfigurable architecture. In: Proc. Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems (CASES). S. 195–204, Oktober 2011.

[He07] Herbordt, M.C.; Van Court, T.; Gu, Yongfeng; Sukhwani, B.; Conti, A; Model, J.; DiSabello, D.: Achieving High Performance with FPGA-Based Computing. IEEE Computer, 40(3):50–57, Marz 2007.

[JM14] Jin, Minxi; Maruyama, Tsutomu: Fast and Accurate Stereo Vision System on FPGA. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 7(1):3:1–3:24, Februar 2014.

[Ke11] Kenter, Tobias; Plessl, Christian; Platzner, Marco; Kauschke, Michael: Performance estimation framework for automated exploration of CPU-accelerator architectures. In: Proc. Int. Symp. on Field-Programmable Gate Arrays (FPGA). ACM, New York, NY, USA, S. 177–180, 2011.

[KSP12] Kenter, Tobias; Schmitz, Henning; Plessl, Christian: Pragma based parallelization – Trading hardware efficiency for ease of use? In: Proc. Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig). IEEE Computer Society, S. 1–6, Dezember 2012.

[KSP15] Kenter, Tobias; Schmitz, Henning; Plessl, Christian: Exploring Trade-Offs between Specialized Dataflow Kernels and a Reusable Overlay in a Stereo Matching Case Study. Int. Journal of Reconfigurable Computing (IJRC), S. 1–24, 2015.

[KVP14] Kenter, Tobias; Vaz, Gavin; Plessl, Christian: Partitioning and Vectorizing Binary Applications for a Reconfigurable Vector Computer. In: Proc. Int. Symp. on Reconfigurable Computing: Architectures, Tools, and Applications (ARC). Springer, April 2014.

[Lö16] Lösch, Achim; Beisel, Tobias; Kenter, Tobias; Plessl, Christian; Platzner, Marco: Performance-centric scheduling with task migration for a heterogeneous compute node in the data center. In: Proc. Design, Automation and Test in Europe Conf. (DATE). 2016.

[Me11] Mei, X.; Sun, X.; Zhou, M.; Jiao, S.; Wang, H.; Zhang, X.: On Building an Accurate Stereo Matching System on Graphics Hardware. In: Proc. ICCV Workshop on GPU in Computer Vision Applications (GPUCV). IEEE, 2011.

[NZ08] Nuzman, Dorit; Zaks, Ayal: Outer-loop vectorization: revisited for short SIMD architectures. In: Proc. Int. Conf. on Parallel Architecture and Compilation Techniques (PACT). PACT '08, ACM, New York, NY, USA, S. 2–11, 2008.

[Sh14] Shan, Yi; Hao, Yuchen; Wang, Wenqiang; Wang, Yu; Chen, Xu; Yang, Huazhong; Luk, Wayne: Hardware Acceleration for an Accurate Stereo Vision System Using Mini-Census Adaptive Support Region. ACM Transactions on Embedded Computing Systems (TECS), 13(4s):132:1–132:24, April 2014.

[Sp09] Spacey, Simon A.; Luk, Wayne; Kelly, Paul H. J.; Kuhn, Daniel: Rapid Design Space Visualization Through Hardware/Software Partitioning. In: Proc. Southern Programmable Logic Conference (SPL). IEEE, S. 159–164, April 2009.

[Ti14] Tippetts, Beau; Lee, Dah Jye; Lillywhite, Kirt; Archibald, James K.: Hardware-Efficient Design of Real-Time Profile Shape Matching Stereo Vision Algorithm on FPGA. Int. Journal of Reconfigurable Computing (IJRC), 2014.

[Va14] Vaz, Gavin; Riebler, Heinrich; Kenter, Tobias; Plessl, Christian: Deferring Accelerator Offloading Decisions to Application Runtime. In: Proc. Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig). IEEE Computer Society, Dezember 2014. Received Best Paper Award.

[Va16] Vaz, Gavin; Riebler, Heinrich; Kenter, Tobias; Plessl, Christian: Potential and Methods for Embedding Dynamic Offloading Decisions into Application Code. Computers & Electrical Engineering, 2016.

Tobias Kenter studierte Informatik an der Universität Paderborn und promovierte ebendort mit Auszeichnung am Lehrstuhl für Hochleistungs-IT-Systeme. In seiner Promotionsarbeit beschäftigte er sich vor allem damit, wie das Effizienzpotenzial rekonfigurierbarer Beschleuniger effektiver für breite Anwendungsfelder erschlossen werden kann. 2014 wurde eine seiner Arbeiten an der ReConFig mit dem Best Paper Award ausgezeichnet. Seit 2016 erforscht Tobias Kenter als Postdoktorand am Paderborn Center for Parallel Computing unter anderem die Skalierbarkeit von parallelen Rechnersystemen mit Beschleunigern. Weiterhin umfassen seine Forschungsinteressen Methoden zur produktiven Nutzung von FPGAs mittels Hochsprachensynthese oder Overlays, Kompromisse zwischen Spezialisierung und Allgemeinheit von Designs, sowie Laufzeitsysteme für Beschleuniger.